In a nutshell: The central challenge of decentralized algorithm design is specifying local rules to achieve desired global behavior. A decentralized algorithm contains four key structures: restrictions, states, events, actions. The design process iterates through specification, analysis, simulation, and testing.

Understand: Nodes in a decentralized system only have access to *local* information about their own state. All other information must be explicitly communicated to them. A decentralized algorithm requires the specification of a *protocol* for interaction between nodes and the operations performed by nodes in response to interactions. Specifically, a protocol comprises:

- *Restrictions*: A list of assumptions about the computational and geographic environment for the protocol, including network connectivity, sensing capabilities, nodes mobility or volatility, uncertainty.
- *Events*: Three different types of (system) events are possible: *message receipt*, *triggers* (e.g., alarms), and *spontaneous* events (external to the system, e.g., node being switched on).
- *Actions*: Actions are "programs": atomic sequences of operations that *may not be interrupted* by events.
- *States*: States enable nodes to retain knowledge of past interactions. Each state and event pair is associated with *exactly* one (possibly empty) action.

Protocol 4.5. Gossiping

Restrictions: $\mathcal{NB}$; gossip probability $g \in [0, 1]$
State Trans. Sys.: $(\{\text{INIT}, \text{IDLE}, \text{DONE}\}, \{(\text{INIT}, \text{DONE}), (\text{IDLE}, \text{DONE})\})$
Initialization: One node in state INIT, all other nodes in state IDLE

```
INIT
  Spontaneously
    broadcast (msge)                    #Broadcast msge to all neighbors
    become DONE

IDLE
  Receiving (msge)
    Randomly select a number n ∈ {0, 1} with probability P(n = 1) = g
    if n = 1 then
      broadcast (msge)                  #Rebroadcast message with probability g
    become DONE
```

Many decentralized spatial algorithms rely on computing inside *spatial structures*, such as boundaries, fronts, regions. A specified protocol must undergo *adversarial analysis*, where designer aims to uncover unfavorable situations or executions that do not perform as expected. Performance may include *correctness*, *robustness* to uncertainty, and *scalability* (primarily communication, but also space and time complexity). After analysis comes simulation and testing. The process iterates until the designer has accumulated convincing evidence that the protocol performs as expected.

Discuss:

- Imagine a specific application of a geosensor network. What would be the expected restrictions on computational and geographic environment?
- How might an adversary disrupt the Gossiping protocol?
- Move to a more complex protocol, Greedy Georouting. Understand the protocol, breaking it down by restrictions, states, events, actions. What can an adversary do to break this protocol?

See: Chapter 3 in Duckham (2012) *Decentralized Spatial Computing*, Springer, Berlin.